

EXHIBIT C

```

package KnowledgeAgents;

import db.*;
import text.*;
5  import utils.*;

import java.io.*;
import java.util.*;

10 /** Repository - Implements the Knowledge Base of a Knowledge Agent */
public class Repository implements Serializable
{
    private transient Agent    agent;          // owning agent
    private transient Trace    trace;          // trace object of owning agent
15  private String            name;            // the name of the repository
    private Profile            profile;         // profile of kept sites
    private SiteDB             keptSites;      // the sites held in the repository
    private int                maxKeptSites;   // max number of kept sites
    private double             historyDecay;
20  private double            freshBlood;

    private transient RepositoryGUI gui;
        transient String[]    initNames;

25  private static final double MIN_HISTORY_DECAY = 0.85;
    private static final double MAX_HISTORY_DECAY = 0.95;
    private static final double DECAY_CHANGE_RATE = 0.99;
    private static final double INITIAL_WEIGHT    = 15.0;

30  // default values
    static final int    FRESH_PERCENT    = 14;
    static final int    MAX_KEPT_SITES  = 50;

    Repository (Agent agent,
35  Trace trace )
    {
        name            = null;
        initNames       = null;
        profile         = new Profile();
40  this.trace         = trace;
        this.agent      = agent;
        keptSites       = new SiteDB(trace,null);
        historyDecay    = MIN_HISTORY_DECAY;

45  gui = new RepositoryGUI(this);
        gui.pack();
        gui.setTitle("Knowledge Agent Initialization");
        gui.setSize(500,400);
        gui.show();
50  }
    /**
     * Initialize some repository values (called by the GUI component)
     * @param The name of the Repository.
     * @param Max. number of sites to keep in the Repository,
55  * @param Max. percent of sites which may refresh the Repository
     * following a query.
     * @param a Stream to the file containing the initial URLs list.
     */
    synchronized final void
60  initValues (String name,
                int    maxKeptSites,

```

```

        int    freshPercent,
        FileInputStream fis )
{
65     this.name          = name;
    this.maxKeptSites = (maxKeptSites > 0) ? maxKeptSites : MAX_KEPT_SITES;
    this.freshBlood    =
        ((freshPercent == 0) ? FRESH_PERCENT : freshPercent) / 100.0;

70     // Now read initial urls list
    if ( fis != null )
    try {
        BufferedReader initFile = new BufferedReader(new InputStreamReader(fis));
        String          urlName;
75         Vector        normNames = new Vector();

        while ( (urlName = initFile.readLine()) != null &&
                urlName.length() > 0
                )
        {
80             String    normName = Filter.normalize(urlName);
            SiteEntry site    = keptSites.add(normName);
            site.historyScore = INITIAL_WEIGHT;
            normNames.addElement(normName);
        }
85         initNames = new String[normNames.size()];
        for ( int i = 0; i < initNames.length; i++ )
            initNames[i] = (String) normNames.elementAt(i);
    }
    catch (IOException e)
90     {
        System.err.println("While reading Initial URLs from file:");
        System.err.println(e);
    }
    finally
95     {
        try {
            fis.close();
        }
        catch(Exception e) {}
100    }

    gui.dispose();
    synchronized(agent) {
        agent.notify();
105    }

}
/**
 * Sets the <code>Trace</code> object of this <code>Repository</code>.
110 * @param The <code>Trace</code> object of the Agent.
 */
public void
setTraceAndAgent (Agent agent,
                  Trace trace)
115 {
    keptSites.setTraceAndGui(trace,null);
    this.agent = agent;
    this.trace = trace;
}

120 static double
avg (double[] array)

```

```

{
    double avg = 0.0;
125     if ( array.length > 0 )
        {
            for ( int i = 0; i < array.length; i++ )
                avg += array[i];
130         avg /= array.length;
        }
    return avg;
}
/**
135  * Returns the Repository's Name.
 */
synchronized public final String
getName()
{
140     return name;
}
/**
 * Expands a query and build an appropriate MiniProfile
 * @param The query to expand.
145  * @param How many LAs to add to each keyword
 * @param The <code>MiniProfile</code> (with weights) for the expanded query.
 * @param The <code>IndexUtil</code> object for this agent.
 * @param The <code>Trace</code> object for this agent.
 * @returns The expanded query.
 */
150 String
expandQuery (String      query,
              int         expandFactor,
              MiniProfile miniProf,
155              IndexUtil indexUtil,
              Trace       trace       )
{
    String xq = indexUtil.expandQuery(query,
                                      expandFactor,
160                                      profile,
                                      miniProf    );
    trace.write("\nXQR Original Query: "+query, Trace.EXPAND_QUERY);
    trace.write( "XQR Expanded Query: "+xq,      Trace.EXPAND_QUERY);
    miniProf.assignWeights(profile);
165     return xq;
}
/**
 * Assigns weights to the terms in a given <code>MiniProfile</code>
 * @param The <code>MiniProfile</code> to which to assign weights.
170  */
final void
assignWeights (MiniProfile miniProf)
{
    miniProf.assignWeights(profile);
175 }
/**
 * Incorporates repository sites into the current <code>SiteDB</code>
 * @param The <code>SiteDB</code> for the current query.
 * @param A vector which will hold added sites.
180  */
final void
incorporate (SiteDB siteDB,
             Vector newSites)

```

```

185 {
    // enumerate the repository sites
    Enumeration sites = keptSites.sites();
    while (sites.hasMoreElements())
    {
        // check if the repository site exists in the current query's siteDB
190 SiteEntry repSite = (SiteEntry) sites.nextElement();
        SiteEntry dbSite = siteDB.add(repSite);
        dbSite.markGroup(SiteEntry.GROUP_KNBASE);
        if ( dbSite.getReadStat() == SiteEntry.READSTAT_NEW )
            newSites.addElement(dbSite.getNormName());
195 }
    }
    /**
     * tranfuses new good sites into the repository, replacing stale sites.
     * @param An array of sites from which to transfuse.
200 * @param Overall scores for the sites, following a query.
     */
    final void
    transfuse (SiteEntry[] sites,
               double[] scores )
205 {
        // first, see how much competition there is on repository seats.
        int maxEntry = (int) (maxKeptSites * freshBlood);
        int sparePlaces = Math.min(maxEntry, maxKeptSites - keptSites.nSites());
        int competitors = maxEntry - sparePlaces;
210
        // arrays for fresh and stale site names
        String[] freshNames = new String[maxEntry];
        String[] staleNames = new String[competitors];
        int nFresh = 0, nStale = 0;
215 int i;

        // initialize fresh and stale names to null
        for ( i = 0; i < maxEntry; i++ )
        {
            freshNames[i] = null;
            if ( i < competitors ) staleNames[i] = null;
        }

        // update the history decay factor
225 historyDecay = Math.min(MAX_HISTORY_DECAY, historyDecay/DECAY_CHANGE_RATE);

        // prepare the two sets of sites (kept/unkept) for scoring
        int[] unkeptInd = new int[sites.length - keptSites.nSites()];
        double[] unkeptScr = new double[unkeptInd.length];
230 int unkept = 0;
        int[] keptInd = new int[keptSites.nSites() + competitors];
        double[] invKeptScr = new double[keptInd.length];
        int kept = 0;

235 for ( i=0; i < sites.length; i++ )
        {
            // update scores of unkept sites
            if ( sites[i].historyScore == 0 )
            {
240 unkeptScr [unkept ] = scores[i];
                unkeptInd [unkept++] = i;
            }
            else
            {

```

```

245      // update the history score of the kept sites.
      SiteEntry keptSite = keptSites.getEntry(sites[i].getNormName());
      keptSite.historyScore = keptSite.historyScore * historyDecay +
          scores[i] * (1.0 - historyDecay);
      invKeptScr[kept ] = -keptSite.historyScore;
250      keptInd[kept++] = i;
    }
  }
  // now get the indices of the best unkept sites.
  int[] bestUnkept = Heap.getBest(unkeptScr, maxEntry);
255  nFresh = sparePlaces;

  if ( competitors > 0 )
  {
    // add competitors to the kept arrays
260    for ( i = sparePlaces; i < bestUnkept.length; i++ )
    {
      // keptInd[kept ] = unkeptInd[bestUnkept[i]];
      invKeptScr[kept++] = -unkeptScr[bestUnkept[i]];
    }
265    // now get the worst sites in this array
    int[] worstKept = Heap.getBest(invKeptScr, competitors);
    /*
     * each kept site in the worst array (= site with low index):
     * 1. Needs to be deleted.
270    * 2. Increases the number of fresh sites that need to be entered.
     */
    for ( i = 0; i < worstKept.length; i++ )
      if ( worstKept[i] < keptSites.nSites() )
      {
275        nFresh++;
        SiteEntry staleSite = sites[keptInd[worstKept[i]]];
        keptSites.removeSite(staleSite.getNormName());
        staleNames[nStale++] = staleSite.getNormName();
      }
280    }

    // nFresh best new sites get into the repository
    for ( i=0; i < nFresh && i < bestUnkept.length; i++ )
    {
285      SiteEntry freshSite = sites[unkeptInd[bestUnkept[i]]];
      freshNames[i] = freshSite.getNormName();
      freshSite.historyScore = unkeptScr[bestUnkept[i]];
      keptSites.add(freshSite);
    }
290

    // we have two lists of sites - new kepts, and ousted kepts.
    agent.updateRepSites(freshNames, true);
    agent.updateRepSites(staleNames, false);

295    if (trace.isLit(Trace.REPOSITORY))
      dumpRepository();
  }

  /**
300  * transfuses new good sites into the repository, replacing stale sites.
  * @param An array of sites from which to transfuse.
  * @param Overall scores for the sites, following a query.
  */
  final void
305  forceAddSites (String[] addNames)

```

```

{
    Filter filter = new Filter(keptSites, SiteEntry.GROUP_KNBASE);
    int i;

310    // enter the required sites into repository, count how many are really new.
    String[] freshNames = new String[addNames.length];
    int actualAdd = 0;
    for ( i = 0; i < addNames.length && actualAdd < maxKeptSites; i++ )
    {
315        addNames[i] = Filter.normalize(addNames[i]);
        if ( filter.forceAddSite(addNames[i]) == Filter.NEW )
        {
            freshNames[actualAdd++] = addNames[i];
        }
320        SiteEntry freshSite = keptSites.getEntry(addNames[i]);
        freshSite.historyScore = INITIAL_WEIGHT;
    }
    for ( i = actualAdd; i < freshNames.length; i++ )
        freshNames[i] = null;
325    // read these sites and update the Repository's profile.
    agent.updateRepSites(freshNames, true);

    // see how many (if any) current repository sites will lose their place.
    int actualDel = Math.max(0, keptSites.nSites() - maxKeptSites);
330    if ( actualDel > 0 )
    {
        // build an array of the current kept sites' score.
        SiteEntry[] keptArray = Agent.getSiteArray(keptSites.nSites(),
                                                    keptSites.sites() );
335        int[] keptInd = new int[keptArray.length];
        double[] invKeptScr = new double[keptArray.length];
        for ( i=0; i < keptArray.length; i++ )
        {
            invKeptScr[i] = -keptArray[i].historyScore;
340            keptInd[i] = i;
        }

        // now get the worst sites in this array, and delete them.
        int[] worstKept = Heap.getBest(invKeptScr, actualDel);
345        String[] staleNames = new String[actualDel];
        for ( i = 0; i < worstKept.length; i++ )
        {
            keptSites.removeSite(keptArray[keptInd[i]].getNormName());
            staleNames[i] = keptArray[keptInd[i]].getNormName();
350        }
        agent.updateRepSites(staleNames, false);
    }
}

355 final void
updateSite (HTMLParse parsed,
            boolean add,
            IndexUtil indexUtil)
{
360    indexUtil.updateProfile(parsed,profile,add);
}

/** dumps the Kept Sites and the Profile to an external file */
private void
365 dumpRepository ()
{

```

```

    try {
        FileWriter file = new FileWriter("./repository.dump");
        file.write ("Printing "+name+" Repository.\n");
370     file.write ("\nNumber of Sites : " + keptSites.nSites()+"\n");

        // enumerate the repository sites
        int      iSite = 0;
        Enumeration sites = keptSites.sites();
375     while (sites.hasMoreElements())
        {
            SiteEntry repSite  = (SiteEntry) sites.nextElement();
            file.write("\nSite # "+iSite+" : " + repSite.getNormName() +
                      " "      + repSite.historyScore);
380     iSite++;
        }

        file.write ("\n\n\nProfile:\n\n");
        profile.print(file);
385     file.close();
    }
    catch (IOException e)
    {
        System.err.println(e);
390     }
    return;
}
}

```